24 / 11 / 2022

Practica 2

Universidad de alicante | Gr. 03 - 3 practicas

Sistemas inteligentes

Adrian Ubeda Touati 50771466R

2022

Contenido

[Parte 1: Aprende las bases de un MLP 2](#_Toc120391518)

[I1) Resolviendo una función booleana mediante MLP 2](#_Toc120391519)

[A) 2](#_Toc120391520)

[B) 5](#_Toc120391521)

[I2) Modelar, entrenar y probar la red en Keras 8](#_Toc120391522)

[A) 8](#_Toc120391523)

[B) 9](#_Toc120391524)

[C) 9](#_Toc120391525)

[D) 11](#_Toc120391526)

[I3) Analizar el entrenamiento y comparar con la red ajustada a mano 12](#_Toc120391527)

[A) 12](#_Toc120391528)

[B) 13](#_Toc120391529)

[C) 14](#_Toc120391530)

[Parte2: Entrena un MLP mediante Deep Learning usando Keras 15](#_Toc120391531)

[II1) Procesamiento de los datos 15](#_Toc120391532)

[A) 15](#_Toc120391533)

[B) 16](#_Toc120391534)

[II2) Implementa la red en keras 20](#_Toc120391535)

[C) 20](#_Toc120391536)

[D) 20](#_Toc120391537)

[II3) Prueba el modelo 22](#_Toc120391538)

[E) 22](#_Toc120391539)

# Parte 1: Aprende las bases de un MLP

## I1) Resolviendo una función booleana mediante MLP

### A)

Texto

Descripción generada automáticamente

La función que tenemos que enfrentar es:



Imagen que contiene naranja, foto, oscuro, cerca

Descripción generada automáticamente

Para el cálculo de los pesos y umbrales, he utilizado Excel.

Aunque parezca una herramienta extravagante, me ha facilitado mucho el calculo de los pesos y umbrales, ya que cuando modificamos un valor, el resto de los valores se modifican al instante sin necesidad de una ejecución.

Primero he implementado la tabla de verdad de cada uno de los términos, siendo estos 5

Tabla

Descripción generada automáticamente

Antes de calcular los pesos me pregunte como seria la estructura de la red neuronal, como sabemos tenemos 5 términos por lo que utilizaremos una neurona para cada termino. Después todos estos términos se juntan con un or, para implementar esta fusión, he puesto una última neurona que se activara si al menos una de 5 neuronas se activa.

Una vez la tabla de verdad echa, he procedido a calcular los pesos para la primera neurona.

He ingresado la siguiente formula en cada fila del primer término:

Interfaz de usuario gráfica, Gráfico, Tabla, Excel

Descripción generada automáticamente

Lo que se busca es que los términos en verde sean positivos y los azules negativos, para cuando se aplique la función sigmoidea, de >0.5 si positivo o <0.5 si negativo.

La fórmula sigue



Siendo w0 el b es decir el peso de la propia neurona, y w\*x la entrada multiplicada por el peso del camino

Y vamos cambiando los pesos hasta conseguir el resultado deseado

Captura de pantalla con letras y números

Descripción generada automáticamente

Captura de pantalla con letras y números

Descripción generada automáticamente

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

### B)

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Imagen que contiene Texto

Descripción generada automáticamente

Para saber si la neurona se activa o no utilizare la función decisión que sigue la formula



Gráfico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ahora hare la función forward:

Texto

Descripción generada automáticamente

Donde para cada nodo, establecemos x como los valores de entrada, w los pesos de cada camino y b el peso de la neurona.

Imagen que contiene Texto

Descripción generada automáticamente

Para probar si la red está bien configurada he creado una función prueba:

En esta función llamo al método forward con cada valor posible, he almacenado en la columna aplicación cada resultado

Tabla

Descripción generada automáticamente con confianza media

Por lo que la red queda verificada

## I2) Modelar, entrenar y probar la red en Keras

### A)

Texto

Descripción generada automáticamente con confianza media

He aprovechado la función prueba echa anteriormente para almacenar las entradas y los resultados en las 2 listas

Texto

Descripción generada automáticamente con confianza media

### B)

Imagen que contiene interior, foto, tabla, cuarto

Descripción generada automáticamente

He modificado un poco el código dando este el siguiente resultado:

Texto

Descripción generada automáticamente

Tenemos 5 nodos entrelazados que reciben 4 valores de entrada que después se asocian con 1, esta red es la que hemos realizado con anterioridad a mano

### C)

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

El batch\_size es el tamaño de la muestra, este tamaño puede tomar como máximo 16 ya que solo tenemos 16 valores para entrenar la red

El número de épocas debe ser el suficiente para que el error de la red sea bajo, probaremos con 2000

También hemos puesto verbose a False para no ver los mensajes generados por el entrenamiento

Antes de ponerlo a entrenar, modificamos el código para que así podamos ver el error que se genera por épocas, usaremos la librería matplotlib para poder representar una curva que nos ayudara a entender el entrenamiento

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Si ejecutamos el código nos genera la siguiente grafica

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Con 2000 epochs, ya conseguiríamos un error relativamente bajo < 0.075

### D)

Texto

Descripción generada automáticamente con confianza media

Gracias a esta parte delco digo vemos los resultados previstos para cada entrada posible:

Texto

Descripción generada automáticamente

Cuando hacemos predict, podemos ver que el resultado corresponde

Texto

Descripción generada automáticamente con confianza media

## I3) Analizar el entrenamiento y comparar con la red ajustada a mano

### A)



Primero veamos la curva generada con el entrenamiento:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Consideremos que la función ha aprendido correctamente si llega al 95% de aciertos es decir <0.05 de error.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Como podemos ver este se consigue por las 1800 – 1900 épocas

Parecen muchas épocas

### B)

***¿Qué es el parámetro loss y optimizer de la red?***



***loss:*** Es el encargado de elegir una función de perdida, esta función de perdida es la encargada de calcular la cantidad que un modelo debería tratar de minimizar durante el entrenamiento, en nuestro caso hemos utilizado mean\_squared\_error

***optimizer:*** Es el encargado de elegir un algoritmo para calcular los pesos, en nuestro caso hemos utilizado Adam

***¿Crees que afectan al entrenamiento?***

El algoritmo de búsqueda de pesos afecta directamente al entrenamiento, ya que, dependiendo de esta, el tiempo de entrenamiento será mayor o menor.

Pasa lo mismo con el manejo de errores, por lo que las 2 variables son cruciales y afectaran enormemente al entrenamiento.

***¿Qué es el learning rate (LR) en una red neuronal?***

Es el proceso en el que la red aprende y progresa para intentar minimizar el error de los resultados

***¿Se puede ajustar el LR en Keras? Haz pruebas para intentar minimizar el número de pasos de entrenamiento requeridos. Comenta los resultados obtenidos***

El LR, depende directamente de la red, y se puede ajustar en Keras, pero cambiando la estructura de la red, además del algoritmo de búsqueda y el manejo de perdidas.

Si queremos minimizar el numero de pasos de entrenamiento requeridos, debemos ir probando diferentes combinaciones de esos 3 factores, en mi caso me enfocare en las capas de nodos.

Antes tenia una capa de 5 y otra que era la salida de 1

Ahora he puesto una capa de 5 otra de 5 una de 3 y una última de 1

Gráfico

Descripción generada automáticamente

Como podemos ver esta curva de aprendizaje es mucho mejor que la anterior necesitando 1100 épocas en vez de 1900 como antes

Gráfico, Gráfico de líneas

Descripción generada automáticamente

### C)

Imagen que contiene naranja, cerca, tabla, pájaro

Descripción generada automáticamente

No se parecen en nada, la estructura de la red ha sido cambiada, además no tiene porque parecerse ya que hay infinitas soluciones, por lo que comparar los pesos no tiene sentido.

# Parte2: Entrena un MLP mediante Deep Learning usando Keras

## II1) Procesamiento de los datos

### A)

Texto

Descripción generada automáticamente

La base de datos MNIST almacena una gran cantidad de imágenes de números escritos manualmente, se utiliza para entrenar la red, como tiene muchos ejemplos, el entrenamiento es muy completo y se consigue una tasa de error muy baja.

Ahora estudiemos la directiva que carga el conjunto:



Analicemos su contenido con el siguiente código:

Texto

Descripción generada automáticamente

Lo que imprime:

Imagen que contiene teclado, electrónica, foto, luz

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

Como podemos ver, el primer argumento tiene 28 listas que cada una de ellas tiene 28 elementos.

Estos hacen referencia a la cantidad de pixeles que hay en la imagen, y tienen como valor un numero del 0 al 255, que corresponde al tono de gris del píxel siendo 255 totalmente negro y 0 totalmente blanco

El 5 impreso en la gráfica, es el resultado de todos esos valores juntos

### B)





En la capa de entrada podríamos asociar un píxel a cada neurona de entrada, en ese caso deberíamos tener 28\*28 = 784 neuronas de entrada y de salida 10, una neurona para con una cadena que simbolice un número entre 0 y 9

Texto

Descripción generada automáticamente



El conjunto de entrenamiento tiene 60000 imágenes y el de test/validación tiene 10000.

Las he recogido en estas 2 tuplas:



X contiene las “imágenes” procesadas y Y los números digitales a los que corresponde

Es importante que el conjunto de entrenamiento sea el mayor posible pero no podemos testear imágenes que han sido entrenadas, por lo que es importante encontrar un equilibrio



Las imágenes primero cambian de formato a una imagen de 28\*28 pixeles, después se vuelve a color binario, es decir, solo blanco o negro



Una vez la imagen preprocesada, tenemos que asociar a cada píxel el número que corresponde su brillo. Después se almacenan en una lista y ya están listas para ser poder realizar el entrenamiento

En el código, hemos tenido que adaptar los valores recogidos, para x\_train y x\_test, hemos tenido que aplanar la lista para pasar de 28 listas de 28 elementos a una sola lista de 784 elementos

Después hemos normalizado el valor para que este entre 0 y 1



Texto

Descripción generada automáticamente

Para y\_test y y\_train, hemos tenido que transformar el numero decimal en una cadena de números que representaran cada posibilidad del 0 al 9

Texto

Descripción generada automáticamente con confianza media

Y después estos valores transformarlos a int32 para que sea compatible en la red neuronal





Una vez la red compilada y entrenada esto es la gráfica que obtenemos:



Texto

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

## II2) Implementa la red en keras

Texto

Descripción generada automáticamente

### C)





La función relu es una función de activación, permite a la red neuronal buscar más allá de las soluciones lineales y así poder encontrar soluciones más complejas.

La función de activación relu, tiene la propiedad que todo valor mayor que 0 conserva su valor, pero todo valor menor que 0 es sustituido por 0



Esta red neuronal es de clasificación, ya que se recoge unos datos, para categorizarlos posteriormente, la función de activación softmax se utiliza en redes neuronales de clasificación puesto que nos da un porcentaje que nos indica la probabilidad de que la imagen pertenezca a una categoría, se usara solo en la última capa

No se podría usar relu puesto que nos devolvería un numero y no un porcentaje



Como el brillo es expresado de 255 a 0, no necesitamos números negativos por lo que con la función relu los eliminamos, después como la función relu es también más eficiente que sigmoidea

### D)





batch\_size: Número de muestras por actualización de gradiente que serán procesadas

validation\_split: es una fracción de los datos de entrenamiento que se usará como datos de validación que fluctúan entre 0 y 1 en tipo flotante



El algoritmo de Backpropagation

Es un algoritmo de ajuste de pesos.

Primero se mira el error en la capa de salida y posteriormente se ajustan los pesos, de igual forma en la capa de salida-1 se mirará el error y se ajustaran nuevamente los pesos

El error en la capa de salida se calcularía de la siguiendo forma:

Un reloj de aguja

Descripción generada automáticamente con confianza media

El error en las capas ocultas se calcularía de esta forma:

Un reloj de aguja

Descripción generada automáticamente con confianza media

Y el peso se recalcularía de la siguiente forma:

Diagrama, Esquemático

Descripción generada automáticamente

## II3) Prueba el modelo



### E)



Texto

Descripción generada automáticamente



Nos que nos da como resultado:





La métrica anterior era la siguiente:



Y esta es la obtenida con la del conjunto entrenamiento



La tasa de error es infinitamente mas pequeña, esto es debido a que la red se ha entrenado con estos valores por los que los tiene interiorizados del entrenamiento, aunque algún ejemplo haya fallado



La métrica anterior era la siguiente:



Y esta es la obtenida con la del conjunto entrenamiento



Como podemos observar se asemejan mucho mas que en el anterior caso, esto es debido a que los valores en los 2 casos no han sido utilizados para el entrenamiento, por lo que la red aun los desconocía.